① 如果 $x=1$，则结点 x 为二叉树的根结点，没有父结点；若 $x>1$，则该结点的父结点编号为 $\lfloor x/2 \rfloor$。

② 如果 $2x>n$，则结点 x 无左子结点，否则结点 x 的左子结点编号为 $2x$；

③ 如果 $2x+1>n$，则结点 x 无右子结点，否则结点 x 的右子结点编号为 $2x+1$。

对于一棵完全二叉树，除最下面一层没有子结点外，其余每一层上的所有结点有且都有两个子结点，这样的完全二叉树就是满二叉树，又称为完美二叉树。

参考词条

1. 二叉树的定义与基本性质
2. 堆排序

延伸阅读

THOMAS H C，CHARLES E L，RONALD L R，et al. 算法导论(原书第 3 版)[M]. 殷建平，徐云，王刚，等译. 北京：机械工业出版社，2013：690-691.

典型题目

NOIP2004 普及组 FBI 树

(佟松龄　叶金毅)

1.3.3.2　完全二叉树的数组表示法

完全二叉树的数组表示法就是使用数组表示一个完全二叉树，其方法是将 n 个结点的完全二叉树，按层次从上到下、从左到右、从 1 开始(根结点编号为 1)依次对每个结点进行编号，并以编号作为数组的下标，依次存放在一维数组中。

图 1.6 是一棵完全二叉树，利用数组表示法，存储在一维数组中，如表 1.10 所示。

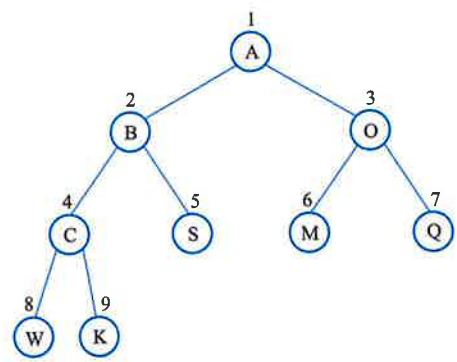

图 1.6　完全二叉树示例

表 1.10　完全二叉树的数组表示法

下标	0	1	2	3	4	5	6	7	8	9	10
存储信息		A	B	O	C	S	M	Q	W	K	

在数组表示法，根据结点存储在数组中的下标数值，可以获得结点在完全二叉树上的父子逻辑关系：

- 非根结点 x(编号 $x>1$)的父结点的编号为 $\lfloor x/2 \rfloor$；
- 结点 x 的左子编号是 $2x$($2x<=n$)，右子编号是 $2x+1$($2x+1<=n$)。

参考词条

1. 堆排序
2. 线段树

延伸阅读

THOMAS H C，CHARLES E L，RONALD L R，et al. 算法导论(原书第 3 版)[M]. 殷建平，徐云，王刚，等译. 北京：机械工业出版社，2013：690-691.

(佟松龄　叶金毅)

1.3.3.3 哈夫曼树的定义、构造和哈夫曼编码

给定 n 个权值作为 n 个叶子结点构造一棵二叉树，使该树的带权路径长度达到最小，则这样的二叉树称为哈夫曼树(Huffman tree)，也称为最优二叉树。

从根结点到某叶子结点经过的边的数量称为该叶子结点的路径长度，每个叶子结点的路径长度与叶子结点权值之积的和称为树的带权路径长度(Weighted Path Length of tree，WPL)，计算公式如下：

$$\mathrm{WPL}=\sum_{i=1}^{n}(W_i\times L_i)$$

其中，n 为叶子结点个数，W_i 为第 i 个叶子结点的权值，L_i 为第 i 个叶子结点的路径长度。哈夫曼树就是 WPL 最小的二叉树。

哈夫曼树构造的基本思想(哈夫曼算法)：权值越大的叶结点越靠近根结点，权值越小的叶结点越远离根结点。

构造的具体过程如下。

(1) 根据给定的 n 个权值$\{w_1,w_2,\cdots,w_n\}$，构造一个森林 $F=\{T_1,T_2,\cdots,T_n\}$。该森林中的每一棵二叉树 T_i 只有权值为 w_i 的根结点，其左、右子树均为空。

(2) 在森林 F 中选取两棵根结点权值最小的树作为左、右子树构造一棵新的二叉树，并且这棵新二叉树根结点的权值为其左、右子树根结点权值之和。

(3) 在森林 F 中删除这两棵二叉树，同时将新得到的二叉树加入森林 F 中。

(4) 重复步骤(2)~(3)，直至森林 F 只包含一棵树为止。最后剩下的这棵二叉树便是所要建立的哈夫曼树。

如设定哈夫曼左子路径编码为 0，右子路径编码为 1，从根结点递归访问每个叶子结点，记录每个根结点到叶子结点路径上的编码，该编码即为哈夫曼编码。

代码示例

构造哈夫曼树的代码如下。

```
const int MAXM = 1010;
const int MAXN = 1010;
```

```
struct HuffmanNode
{
    double weight;
    int id,fa,lc,rc; //结点 id 的权值为 weight,结点 id 的父结点为 fa,结点 id 的左右
                     孩子分别为 lc 和 rc
    bool operator<(const HuffmanNode &a) const
    { // weight 小优先
        return weight > a.weight;
    }
} ht[MAXM];
double w[MAXN]; //数组 w 中存放着 n 个叶子结点的权,构造哈夫曼树 ht。
bool HuffmanTree(int n)
{
    if (n <= 1)
        return 0;
    int m = 2 * n - 1;
    // m 记录哈夫曼树的结点个数,哈夫曼树没有度为 1 的结点,对于具有 n 个叶子结点的哈夫
       曼树,根据二叉树的性质,可以推出其具有 2n-1 个数组元素
    priority_queue<HuffmanNode> hq;
    for (int i = 1; i <= n; i++)
    { //执行哈夫曼算法的第 1 步
        ht[i].weight = w[i];
        ht[i].id = i;
        ht[i].fa = 0;
        ht[i].lc = ht[i].rc = 0;
        hq.push(ht[i]);
    }
    HuffmanNode s1,s2;
    for (int i = n + 1; i <= m; i++)
    { //重复执行哈夫曼算法的第 2~3 步,直至只剩一棵树
        s1 = hq.top();
        hq.pop();
        s2 = hq.top();
        hq.pop();
        ht[i].weight = s1.weight + s2.weight;
        ht[i].id = i;
        ht[i].fa = 0;
        ht[i].lc = s1.id;
        ht[i].rc = s2.id;
        ht[s1.id].fa = i;
        ht[s2.id].fa = i;
        hq.push(ht[i]);
    }
    return 1;
}
```

参考词条

1. 二叉树的定义与基本性质
2. 完全二叉树的定义与基本性质

延伸阅读

[1] THOMAS H C，CHARLES E L，RONALD L R，et al. 算法导论(原书第 3 版)[M]. 殷建平，徐云，王刚，等译. 北京：机械工业出版社，2013：245-249.
[2] 严蔚敏，吴伟民. 数据结构[M]. 北京：清华大学出版社，2007：144-149.

典型题目

NOI2015 荷马史诗

(佟松龄　叶金毅)

1.3.3.4　二叉搜索树的定义和构造

二叉搜索树(Binary Search Tree，BST)又称为二叉排序树、二叉查找树。一棵二叉搜索树或者是空树，或者是结点包含权值，且具有以下特征：

(1) 若左子树不空，则左子树上所有结点的权值均小于根结点的权值；
(2) 若右子树不空，则右子树上所有结点的权值均大于根结点的权值；
(3) 左、右子树也是二叉搜索树。

二叉搜索树一般有查找、插入、删除等操作。

1. 查找操作

在以 p 为根结点的二叉搜索树中查找一个权值为 val 的结点，步骤如下：

(1) 若 p 为空，则查找不成功；
(2) 若 p 的权值等于 val，则查找成功；
(3) 若 val 小于 p 的权值，则递归查找左子树；
(4) 若 val 大于 p 的权值，则递归查找右子树。

2. 插入操作

在以 p 为根结点的二叉搜索树中插入一个权值为 val 的新结点，步骤如下：

(1) 若 p 为空，直接插入新结点；
(2) 若 p 的权值大于 val，在 p 的左子树中插入权值为 val 的结点；
(3) 若 p 的权值小于 val，在 p 的右子树中插入权值为 val 的结点。

3. 删除操作

在以 p 为根结点的二叉搜索树中删除一个权值为 val 的结点，步骤如下。

如果查找到权值为 val 的结点，分三种情况进行处理：

(1) 若该结点的度为 0，即为叶子结点，可以直接删除。
(2) 若该结点的度为 1，即只有左子树或右子树，则用这棵子树代替 p 结点即可。

(3) 若该结点的度为 2，即左子树和右子树均不空，有两种不同做法。

方法 1：找出左子树中权值最大的结点 x，将该结点的权值替换为 x 结点的权值，再按照(2)或(3)方法删除 x 结点(x 作为左子树中权值最大的结点，度一定是 0 或 1)。

方法 2：找出右子树中权值最小的结点 y，将该结点的权值替换为 y 结点的权值，再按照(2)或(3)方法删除 y 结点(y 作为右子树中权值最小的结点，度一定是 0 或 1)。

二叉搜索树的构造方法：将给定的所有结点，依次按照插入操作的步骤，插入到二叉搜索树中相应的叶结点位置，即完成了二叉搜索树的构造。

二叉搜索树的遍历与二叉树的遍历方式相同。对二叉搜索树中序遍历，可以得到各结点权值从小到大排序的序列。

代码示例

二叉搜索树的构造、查找、插入、删除、中序遍历代码如下。

```
struct BSTNode
{
    int lc,rc; // 左、右子结点在数组中的下标
    int val;// 值
} BST[SIZE];// 数组模拟链表存储二叉排序树
int tot = 0;//总结点数
int root,INF = 1 << 30;
//在二叉排序树中建立一个新结点
int NewNode(int val)
{
    BST[++tot].val = val;
    return tot;
}
//建树初始化
void InitBuild()
{
    NewNode(-INF);
    NewNode(INF);
    root = 1;
    BST[1].rc = 2;
}
//在二叉排序树中从结点 p 开始,查找值为 val 的结点,并返回结点编号
int SearchBST(int p,int val)
{
    if (p == 0) // 查找失败
        return 0;
    if (val == BST[p].val) // 查找成功
        return p;
    else if (val < BST[p].val)
```